(12) DEMANDE INTERNATIONALE PUBLIÉE EN VERTU DU TRAITÉ DE COOPÉRATION EN MATIÈRE DE BREVETS (PCT)

(19) Organisation Mondiale de la Propriété Intellectuelle
Bureau international

(43) Date de la publication internationale
25 mars 2004 (25.03.2004)

PCT

(10) Numéro de publication internationale
WO 2004/025507 A2

(51) Classification internationale des brevets⁷ : G06F 17/30

(21) Numéro de la demande internationale :
PCT/FR2003/002675

(22) Date de dépôt international :
9 septembre 2003 (09.09.2003)

(25) Langue de dépôt : français

(26) Langue de publication : français

(30) Données relatives à la priorité :
02/11250 11 septembre 2002 (11.09.2002) FR

(71) Déposant *(pour tous les États désignés sauf US)* : **KARMIC SOFTWARE RESEARCH** [FR/FR]; 18, rue Jacquemont, F-75017 Paris (FR).

(72) Inventeur; et

(75) Inventeur/Déposant *(pour US seulement)* : **ZAM-FIROIU, Michel** [FR/FR]; 18, rue Lisfranc, F-75020 Paris (FR).

(74) Mandataires : **BREESE, Pierre** etc.; Breesé-Majerowicz, 3, avenue de l'Opéra, F-75001 Paris (FR).

(81) États désignés *(national)* : AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) États désignés *(régional)* : brevet ARIPO (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), brevet eurasien (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), brevet européen (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), brevet OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

[Suite sur la page suivante]

(54) Title: METHOD FOR ORGANIZING A DIGITAL DATABASE IN A TRACEABLE FORM

(54) Titre : PROCEDE D'ORGANISATION D'UNE BASE DE DONNEES NUMERIQUES SOUS UNE FORME TRAÇABLE

(57) Abstract: The invention concerns a method for organizing a digital database in a traceable form, comprising steps which consist in modifying a main digital database by adding or suppressing or changing a record of the main base and steps which consist in reproducing the main database. The invention is characterized in that the step of modifying the main database includes an operation which consists in creating at least one digital recording comprising at least: the unique digital identifiers of the relevant records and attributes of the main database, a unique digital identifier of the state of the main database corresponding to said modification of the main database, the elementary values of the attributes assigned to them through the elementary operations, without storing the unmodified attributes and records, and adding said record in an internal historization base consisting of at least one table, and in that the reproduction step concerning any final or prior state of the main database consists in receiving (or intercepting) an original request associated with the identifier of the state concerned, in transforming said original request to constitute a modified addressing request of the historization base comprising the criteria of the original request and the identifier of the state concerned, and reconstituting the record(s) corresponding to the criteria of the original request and to the state concerned, the method further comprising mechanisms enabling the monitoring of the operations, the causal dependencies and the impact of the modifications, as well as the fusion of branches or the evolution of the data structure.

(57) Abrégé : Procédé d'organisation d'une base de données numériques sous une forme traçable, comportant des étapes de modification d'une base de données numériques principale par ajout ou suppression ou modification d'un enregistrement de la base principale et des étapes de lecture de la base de données principale, caractérisé en ce que l'étape de modification de la base de données principale comprend une opération de création d'au moins un enregistrement numérique comportant au moins: les identifiants numériques uniques des enregistrements et des attributs concernés de la base de données principale, un identifiant numérique unique de l'état de la base de données principale correspondant à ladite modification de la base de données principale, les valeurs élémentaires des attributs qui leur sont affectées à travers les opérations élémentaires, sans procéder au stockage des attributs ou des enregistrements non modifiés, et d'ajout dudit enregistrement dans une base d'historisation interne composée d'au moins une table, et en ce que l'étape de lecture portant sur tout état final ou antérieur de la base de données principale consiste à recevoir (ou intercepter) une requête originelle associée à l'identificateur unique de l'état visé, à procéder à une transformation de ladite requête originelle pour construire une requête modifiée d'adressage de la base d'historisation comprenant les critères de la requête originelle et l'identificateur de l'état visé, et de reconstruction du ou des enregistrements correspondant aux critères de la requête originelle et à l'état visé, le procédé comprenant en outre des mécanismes destinés à permettre le suivi des opérations, des dépendances causales et l'impact des changements, ainsi que la fusion de branches ou l'évolution de la structure des données.

WO 2004/025507 A2

Publiée :

— *sans rapport de recherche internationale, sera republiée dès réception de ce rapport*

En ce qui concerne les codes à deux lettres et autres abréviations, se référer aux "Notes explicatives relatives aux codes et abréviations" figurant au début de chaque numéro ordinaire de la Gazette du PCT.

PROCEDE D'ORGANISATION D'UNE BASE DE DONNEES NUMERIQUES SOUS UNE FORME TRAÇABLE

La présente invention se rapporte au domaine de la gestion des données persistantes d'une entité, par exemple une entreprise. En particulier, la présente invention se rapporte au suivi de ces données persistantes dans une base de données par l'intermédiaire d'un Système de Gestion de Base de Données. Il est en effet difficile pour une entreprise de garantir le suivi du processus d'évolution des données persistantes stratégiques car ce suivi présente quelques obstacles objectifs :

- Caractère asynchrone et collaboratif du déroulement du processus
- Caractère très exigeant du suivi pour constituer une réelle garantie : la présence d'un maillon faible compromet définitivement la fiabilité de toute réponse
- Non disponibilité de solutions génériques de prise en charge de la traçabilité dans les couches logicielles du marché à un niveau de granularité satisfaisant : OS, SGBD, langage de développement
- Coût très élevé de récriture des applications existantes et coût très élevé de prise en compte explicite de la traçabilité par chaque application.

L'art antérieur connaît déjà par la demande de brevet international WO 9935566 un procédé d'identification et de suivi des évolutions d'un ensemble de composants logiciels. Le procédé proposé par ce document de l'art antérieur permet de recenser des composants par leur nom et leur version. Cette classification au niveau fichier ne correspond pas au problème de conserver des traces de données de manière continue, c'est-à-dire à chaque modification desdites données. En particulier, le procédé

proposé ne convient pas pour tracer une base de données modifiée à chaque accès en écriture.

Il est proposé, dans le brevet américain US 5,347,653 une méthode fournissant une perspective historique à une base de données d'objets grâce à un versionnement des objets stockés ainsi qu'à une indexation représentative des objets. Cette méthode de l'art antérieur propose de stocker intégralement la dernière version de la base de données et de stocker d'autre part les différences à appliquer à cette dernière version pour obtenir des versions antérieures. Le problème posé par ce document est la nécessité d'appliquer les différences une à une et en série pour trouver l'état de la base à une date donnée. Cette contrainte implique un coût en temps important.

L'art antérieur connaît également, par la demande de brevet PCT WO 02/27561 (Oracle), un système et un procédé pour fournir un accès à une base de données temporelle. L'invention décrite dans ce document concerne un système et un procédé de visualisation sélective de données en rangées temporaires dans une base de données de lecture constante. Les transactions sauvegardées provoquant des changements dans les données en rangées d'une base de données sont pistées et un numéro de changement du système stocké est assigné à chaque transaction sauvegardée. Une sélection demandée de valeurs de données en rangées de la base de données est exécutée ainsi qu'un temps d'interrogation ayant lieu avant le temps de sauvegarde d'au moins une transaction sauvegardée. Les valeurs des données en rangées ordonnées contenues dans les segments d'annulation stockant un identificateur de transaction pour au moins une transaction sauvegardée sont récupérées.

L'art antérieur connaît également, par la demande de brevet PCT WO 92/13310 (Tandem Telecommunications Systems), un procédé de sélection et de représentation de données variant dans le temps à partir d'un système de gestion d'une base de données évoluant en fonction du temps, ledit procédé produisant une vue unifiée sur un écran d'ordinateur. Les données provenant d'un enregistrement maître relatif à une entité particulière sont affichées avec un attribut vidéo ou de caractère par défaut, et sont considérées comme étant l'enregistrement à jour. L'accès à un enregistrement historique relatif à cette entité fait que les données relatives à des champs qui diffèrent des champs correspondants de l'enregistrement à jour sont superposées à de tels champs d'enregistrement à jour mais avec un attribut vidéo ou de caractère différent de l'attribut vidéo ou de caractère par défaut. L'enregistrement à jour superposé devient un nouvel enregistrement à jour destiné à d'autres superpositions. De la même façon, l'accès à un enregistrement en suspens fait que les données relatives à des champs qui diffèrent des champs correspondants de l'enregistrement à jour sont superposées à de tels champs d'enregistrement à jour mais avec un attribut vidéo ou de caractère différent de l'attribut vidéo ou de caractère par défaut. Une pluralité d'enregistrements historiques ou en suspens peuvent être composés de sorte que tous les champs modifiés pour un jeu d'enregistrement à partir de la fin d'une période définie peuvent être superposés à un enregistrement à jour en une seule fois.

On connaît également, par la demande de brevet européen EP 0 984 369 (Fujitsu), un mécanisme de stockage de versions datées de données. Dans ce mécanisme de stockage, les données sont stockées comme une pluralité d'enregistrements, chaque enregistrement comportant au

moins un attribut, un marqueur de temps indiquant la durée pour laquelle l'attribut est valide, un temps d'insertion indiquant le moment où l'enregistrement a été créé et un champ de type. Le champ de type indique si l'enregistrement est un enregistrement concret, un enregistrement delta, ou un enregistrement d'archive remplaçant un ou plusieurs enregistrements qui ont été archivés. On accède aux données pour trouver une valeur d'attribut du point de vue d'un « temps spécifié », en réalisant une extraction des enregistrements qui possèdent des temps d'insertion antérieurs au « temps spécifié » et en construisant une valeur d'attribut à partir des enregistrements extraits. Les données sont mises à jour uniquement en ajoutant des enregistrements concrets ou des enregistrements delta, sans modifier les valeurs d'attribut dans les enregistrements concrets ou les enregistrements delta.

La présente invention entend remédier aux inconvénients de l'art antérieur en proposant un procédé de suivi de l'évolution des données dans une architecture basée sur un SGBD, consistant en :

- la matérialisation des versions intermédiaires et des flux de données résultantes des opérations effectuées sur la base de données, au fur et à mesure de son évolution, au niveau de granularité élémentaire (enregistrement par enregistrement et attribut par attribut) ;
- la possibilité de reconstitution et restitution « rapide » de tout état cadre historique d'origine de chaque version de donnée et chaque opération (par « rapide » nous comprenons « sans temps additionnel perceptible lié à la restauration ») ;

comprenant :

- des mécanismes de reconstitution de flux de dépendance causale (de type source-destination) entre les données concernées ;
- des mécanismes de notification de remise en cause des opérations du passé en cas d'évolution des données d'entrée ;
- des mécanismes de ré-exécution ;

et couvrant les cas particuliers et les extensions suivants :

- prise en compte de l'évolution structurelle (évolution de schéma) ;
- prise en compte de l'évolution des applications ;
- prise en compte des applications existantes dans un cadre architectural flexible ;
- schémas d'évolution graduelle d'une architecture à l'échelle de l'entreprise ;
- gestion de versions virtuelles (familles alternatives et hypothèses parallèles).

Le but principal de l'invention est de permettre l'exploitation des données de la base selon des versions successives, tout en limitant les besoins de temps et capacité de stockage et à autoriser la restitution à la volée.

Une démarche habituelle consiste à enregistrer des versions successives des bases de données, par exemple sous la forme de stockage périodique sur un support telle qu'une cartouche magnétique de l'intégralité de la base de données, dans l'état correspondant à la version courante. La recherche d'une information nécessite la restauration

préalable de toute la base, à partir du support correspondant à la sauvegarde correspondante, puis à interroger la base ainsi restaurée. Pour des bases de données importantes telles qu'exploitées dans le domaine bancaire, de l'assurance ou de la gestion, le volume correspondant à un état peut dépasser le téraoctet, volume qu'il convient de multiplier par le nombre d'états sauvegardés.

Cette solution est totalement inadaptée à l'exploitation en temps réel.

L'invention vise à répondre au problème technique de l'exploitation en temps réel de bases de données de grande volume.

A cet effet, l'invention concerne dans son acception la plus générale un procédé d'organisation d'une base de données numérique sous une forme traçable, comportant des étapes de modification d'une base de données numériques principale par ajout ou suppression ou modification d'un enregistrement de la base principale et des étapes de lecture de la base de données principale,

caractérisé en ce que

l'étape de modification de la base de données principale comprend une opération de création d'au moins un enregistrement numérique comportant au moins :

les identifiants numériques uniques des enregistrements et des attributs concernés de la base de données principale,

un identifiant numérique de l'état de la base de données principale correspondant à ladite modification de la base de données principale,

les valeurs élémentaires des attributs qui leur sont affectées à travers les opérations élémentaires, sans procéder au stockage des attributs ou des enregistrements non modifiés,

et d'ajout dudit enregistrement dans une base d'historisation interne composée d'au moins une table d'historisation interne,

et en ce que l'étape de lecture portant sur tout état final ou antérieur de la base de données principale consiste à recevoir (ou intercepter) une requête originelle associée à l'identificateur unique de l'état visé, à procéder à une transformation de ladite requête originelle pour construire une requête modifiée d'adressage de la base d'historisation comprenant les critères de la requête originelle et l'identificateur de l'état visé, et de reconstruction du ou des enregistrements correspondant aux critères de la requête originelle et à l'état visé, ladite étape de reconstitution consistant à retrouver les valeurs élémentaires, contenues dans les enregistrements de la base d'historisation, correspondant aux critères de la requête originelle [afin de réduire les besoins de capacité de stockage et les temps de traitement].

Selon une variante, lesdits enregistrements de la base de données d'historisation contiennent également des références à d'autres enregistrements de la base de données interne, dans le but de préciser les liens de dépendance dynamique de type source-destination constituant le flux causal des interférences entre les versions des données.

Avantageusement, ladite opération de modification de la base principale est une opération logique et ladite opération d'ajout dans la base de données d'historisation consiste à ajouter :

un enregistrement identifiant l'état de la base correspondant à l'opération logique,

autant d'enregistrements que de paramètres de l'opération logique,

un enregistrement pour le résultat éventuel de l'opération logique

et à préciser par un lien de parenté les regroupements d'opérations depuis le niveau élémentaire de modification jusqu'au niveau de la transaction, en passant le nombre de niveaux sémantiques nécessaires pour les applications.

Selon une autre variante, la base de données principale contient une ou plusieurs table(s), organisant les liens d'évolution entre les identifiants des états successifs et alternatifs de la base principale, destinée(s) à organiser les enregistrements de la base de données interne.

De préférence, ladite ou lesdites tables des liens d'évolution entre les états de la base principale contiennent des enregistrements spécifiant les règles de correspondance entre les enregistrements de la base de données interne d'historisation et les états de la base de données principale.

Selon un mode de mise en œuvre particulier, ladite opération de lecture consiste à déterminer ledit état de la base de données principale en se référant aux dits identifiants et aux tables des liens d'évolution entre les états de la base principale.

Avantageusement, une application interrogeant la base de données principale peut spécifier l'état de la base de données principale désiré.

L'invention concerne également une architecture de gestion de base de données caractérisée en ce que ladite application peut opérer des modifications sur tout état de la base principale et donnant lieu, dans le cas de la tentative de modification d'un état antérieur, à la création de nouvelles alternatives d'évolution de la base

de données principale dont les données seront générées par la même base d'historisation interne.

Selon une variante, les liens de dépendances servent de critères de remise en cause desdites opérations déjà effectuées.

De préférence, les mises à jour effectuées sur des branches différentes pourront être intégrées ou fusionnées dans le cadre d'un nouvel état « héritant » desdites branches.

Selon un mode de mise en œuvre particulier, les cas d'évolution de structure des données de la base de données principale sont traités comme des cas particuliers d'évolution des données de ladite base, pour peu que la structure/schéma de ladite base principale soit décrite de la façon mentionnée pour les données, en tant que dictionnaire.

Selon une autre variante, la base de données d'historisation est explorée et interrogée par des applications à travers le mode natif du SGBD afin d'obtenir des informations comme par exemple toutes les valeurs historiques d'un attribut et toutes les incidences (dynamiques) de toute mise à jour et de naviguer au long des versions et des flux de dépendance dynamique et ceci de façon classique, selon le langage d'interrogation en vigueur, exigé par le SGBD.

On comprendra mieux la présente invention à l'aide de la description, faite ci-après à titre purement explicatif, d'un mode de réalisation de l'invention, en référence aux figures annexées :

La figure 1 illustre une architecture classique de communication entre une application et une base de données ;

La figure 2 illustre une architecture de communication similaire à celle de la figure 1 et comprenant les éléments nécessaires à l'application de l'invention ;

La figure 3 illustre les différents moyens d'accès à une base de données organisée de façon traçable munie d'un système selon l'invention.

La gestion des données persistantes d'une entreprise (ou d'une organisation au sens large) est généralement confiée à un logiciel spécifique appelé aussi SGBD. Les applications informatiques proposent aux utilisateurs des moyens ergonomiques interactifs capables de visualiser et faire évoluer les données de la base de données de l'entreprise en communiquant avec le SGBD. Dans les paragraphes suivants, nous rappelons les principales caractéristiques de l'architecture afin de positionner le cadre de notre procédé de suivi de l'évolution des données et d'en fixer le vocabulaire minimal.

Le gestionnaire de persistance nécessaire pour notre système autorise le stockage des données et leur reconstitution en mémoire en conformité avec leur structure (définie comme un ensemble d'attributs) et les valeurs saisies ou calculées. Les principaux SGBD Relationnels (mais aussi bien de type objet, réseau ou hiérarchiques) du marché sont des bons candidats pour le rôle de gestionnaire de persistance. Cette compatibilité est d'ailleurs un atout de notre procédé qui peut aussi tirer ainsi profit de la base logicielle installée dans l'entreprise.

Considérons par simplification - et uniquement à titre d'exemple - l'utilisation d'un SGBD Relationnel. Celui-ci permet la représentation des données sous forme de tables (ou relations). Les colonnes indiquent les attributs (ou champs). Chaque colonne est caractérisée par un domaine

(entier, caractère, date, flottant, etc.) et d'autres informations éventuelles comme la taille maximale (pour les chaînes de caractères). Certains attributs (un ou plusieurs) constituent la clé ou l'identifiant de l'enregistrement. Dans la figure suivante, nous avons représenté une table et nous avons indiqué les clés en mode souligné. Chaque ligne d'une même table représente un nouvel enregistrement (ou n-uplet) de structure uniforme. Chaque cellule représente la valeur de l'attribut. Par exemple, « aaa » est la valeur de l'attribut Attribut1 du premier enregistrement dont la clé est 1001.

Table

Clé	**Attribut1**	**Attribut2**
1001	« aaa »	23/12/2001
1002	« bbb »	24/11/2000
1003	« ccc »	8/05/1989

Les données sont insérées, lues, modifiées et supprimées à travers un langage de manipulation de données (SQL par exemple).

Le gestionnaire de persistance permet également la définition, la consultation et l'évolution de la structure des données, appelée aussi schéma de données. Ainsi, les tables peuvent être définies, supprimées ou restructurées. Dans le dernier cas, des colonnes peuvent être rajoutées ou supprimées. Parfois, il est utile même de changer le domaine d'un attribut, ou d'autres caractéristiques analogues, ce qui peut impliquer des traitements implicites ou explicites de conversion des données concernées.

Quelle que soit la représentation physique des données, la table est la référence logique de

représentation des données. Ainsi, les applications « voient » généralement les données sous la forme de tables. Il est important de souligner que notre système tient à préserver cette représentation logique afin de s'assurer la plus grande compatibilité avec les applications existantes. Par exemple, après avoir demandé la connexion à une base de données particulière, une application peut s'adresser à un gestionnaire de persistance avec une requête de type "select * from client" et recevoir en échange un ensemble de données permettant la reconstitution des données sous forme tabulaire.

Précisons enfin qu'une base de données représente un état cohérent du monde réel représenté. Les données de la base évoluent par à-coups déclenchés par des événements à travers des opérations (insertion, mise à jour ou suppression) regroupées généralement en transactions. Ces dernières sont caractérisées par des propriétés particulières dites ACID (Atomicité, Cohérence, Isolation et Durabilité) qui garantissent un certain niveau de qualité.

Assurer la traçabilité des données persistantes revient à fournir des moyens permettant le suivi en amont et en aval du processus d'évolution des données.

Le processus d'évolution des données est une suite généralement non prédictible d'exécutions d'opérations élémentaires qui lisent, transforment et écrivent les données de façon répétée, donnant lieu le plus souvent à des interférences multiples et complexes qui rendent difficile et souvent impossible leur suivi. Assurer la traçabilité du processus revient à être capable de remonter à tout moment vers les origines (débuts) du processus, retrouver les valeurs des données d'origine,

pouvoir suivre et comprendre au fil des opérations leurs conséquences en termes d'impact de changements. En termes de qualité de l'information, la traçabilité est très précieuse car elle permet de garantir la conformité du résultat d'une opération appliqué avec le jeu de données d'entrée.

Pour mieux comprendre l'étendue de sa portée, nous présentons une classification de la traçabilité selon des niveaux de plus en plus avancés :

- le premier niveau de traçabilité, que l'on peut qualifier d'élémentaire, est celui de la représentation et du stockage des données. Il s'agit donc de décrire la structure, puis de stocker et d'identifier la donnée, qu'il s'agisse d'une commande, d'un article ou encore d'une pièce mécanique, afin de pouvoir la retrouver plus tard. Ce type de fonctionnalité est déjà assuré par des logiciels spécialisés, appelés Systèmes de Gestion de Bases de Données (SGBD). Le processus d'évolution se manifeste par l'application successive d'opérations élémentaires comme la lecture, l'insertion, la mise à jour et la suppression. Ces opérations élémentaires sont généralement regroupées en transactions afin de maintenir la cohérence des données dans des conditions d'utilisation concurrente ou encore de reprise sur panne. A ce niveau, les mises à jour ont comme conséquence naturelle la perte des valeurs existantes suite à leur remplacement par des nouvelles valeurs, puisque - par convention - à un même identifiant ne peut correspondre qu'une seule donnée (avec ses attributs). Ce premier niveau dit élémentaire de traçabilité est indispensable mais largement insuffisant.

- le second niveau de traçabilité autorise une donnée à disposer en même temps de plusieurs versions (valeurs distinctes). Cela améliore la traçabilité puisqu'il devient possible de disposer à tout moment aussi bien des valeurs précédant que de celles suivant l'exécution d'une opération ou d'un processus, ce qui facilite davantage la compréhension de l'évolution. Le versionement introduit une qualité précieuse, puisque l'irréversibilité n'est plus incontournable (on permet l'évolution des données, sans perte des valeurs actuelles). En plus des versions successives, il existe des versions alternatives. Il arrive souvent qu'un utilisateur - après avoir remonté le fil d'exécution d'un processus - souhaite opérer quelques changements sur l'état antérieur des données. Dans ces cas, les mécanismes de versionement permettent la prise en compte d'alternatives, ou des branches d'évolution qui autorisent plusieurs suites possibles à partir d'un même état de la base. Un système de traçabilité avancé doit donc intégrer cet aspect, d'autant plus qu'une nouvelle branche permet de ne pas détruire les précédentes et de préserver ainsi la traçabilité des processus antérieurs. Il existe des nombreux travaux qui prennent en compte les données dont les valeurs évoluent dans le temps. Le domaine des bases de données temporelles distingue clairement l'axe du temps de validité de celui du temps de transaction. Le temps de validité permet de préciser par exemple qu'un tarif est valable de telle à telle date. Cette information est totalement indépendante de la date de la mise à jour de la donnée qui la stocke dans la base et qui se situe dans le temps dit transactionnel. De par la nature spécifique de leurs problématiques, les mécanismes de prise en compte du temps de validité comprennent des solutions

d'interrogation et de mise à jour [Publication de R. Snodgrass « The temporal query language Tquel » ACM Transactions on database Systems, Association for Computer Machinery. New York, USA], proposent des opérateurs dédiés à la prise en compte d'intervalles (between, before, etc), et traitent spécifiquement les cas de mise à jour d'intervalles de temps pour une même donnée qui impliquent la fusion ou la division [Demande de brevet européen EP 0 984 369 (Fujitsu)]. Par ailleurs, la représentation et l'affichage des différentes versions réclament à leur tour des solutions spécifiques [Demande de brevet PCT WO 92/13310 (Tandem Telecommunications Systems)] qui facilitent la compréhension de l'évolution de données individuelles, sans se préoccuper de branches ou encore du critère global de cohérence collective des données de la base dans l'espace de versionement. En effet, ces aspects se situent en dehors de la problématique de traçabilité qui maintient vis-à-vis du versionement une série d'exigences qui lui sont propres et qui restent toujours non résolues. Citons enfin l'archivage et la restauration comme mécanismes permettant de retrouver des états antérieurs de la base de données. Il est évident en revanche leur inadéquation face à la problématique de traçabilité, pour des raisons de granularité trop grossière de suivi de l'évolution engendrant des inconvénients insolubles de temps de réponse et d'espace de stockage. En conclusion, le versionement est également indispensable pour assurer la traçabilité, mais reste, comme nous le verrons plus bas, toujours insuffisant.

- un troisième niveau de traçabilité est celui des opérations. Tracer une opération revient à laisser une

trace persistante de l'exécution de ladite opération, permettant une encore meilleure compréhension de la façon dont les données évoluent. On peut ainsi mieux expliquer l'évolution d'une commande entre deux versions, si l'on sait par exemple qu'il y a eu une opération de remise sur le prix total. La plupart des SGBD disposent de mécanismes de journalisation qui autorisent la consultation des opérations effectuées au niveau élémentaire. Pour qu'elles soient compréhensibles par les utilisateurs, ces informations doivent être corrélées avec les opérations de haut niveau, or le problème de fond est que les entrées du journal n'ont pas le même cycle de persistance que les données. Ainsi, le journal se trouve généralement en dehors de la base de données et se voit régulièrement purgé par l'administrateur. La demande de brevet PCT WO 02/27561 (Oracle) apporte une solution alternative à ce problème, en proposant le stockage interne (dans la base de données) des transactions et des informations d'annulation de leurs effets (undo), ce qui permet de retrouver tout état antérieur de la base de données par l'exécution dans l'ordre inversé de l'inverse des opérations qui ont eu lieu depuis. Bien qu'intéressante, cette technique peut être très onéreuse en termes de temps d'exécution car, pour retrouver une version précise d'une donnée, elle défait toutes les opérations qui ont lieu depuis, y compris celles qui ne la concernent pas. De plus, elle n'est pas appropriée non plus pour obtenir la liste de toutes les versions d'une donnée. Enfin, elle interdit toute mise à jour à partir d'un état antérieur de la base, ce qui écarte les variantes et les branches alternatives d'évolution. Comme nous le verrons plus loin, dans la présente invention, les inventeurs ont opté pour une stratégie

opposée : à la réception d'une requête on procède dans la présente invention à la transformation de celle-ci puis à une exécution sur les données versionées. Notons enfin, la nécessité de disposer d'informations de plus haut niveau, fournies par exemple par les applications, afin d'obtenir une articulation entre la sémantique des applications (application d'une remise sur une commande) et celle du SGBD (mise à jour de l'attribut « montant » de la commande).

- le niveau le plus avancé de la traçabilité est celui de la causalité. Il vise la matérialisation des liens de transport d'information au niveau le plus élémentaire (le grain le plus fin). Par exemple, si une opération quelconque O procède à la lecture de l'attribut A de la donnée X, à la lecture de l'attribut B de la donnée Y, à l'addition des deux et au stockage de la valeur ainsi obtenue dans l'attribut C de la donnée Z, un lien causal serait capable de reconstituer ce transport d'information à travers les différentes versions des données X, Y et Z, ainsi qu'aux diverses exécutions de l'opération O. Cette précieuse information permet de comprendre les détails des évolutions, d'expliquer transitivement les origines des modifications et de détecter les opérations qui sont à refaire en cas d'évolution des données d'origine. Elle est surtout importante parce que - contrairement aux techniques de journalisation - elle se défait de la contrainte séquentielle des opérations pour se concentrer sur les dépendances dynamiques engendrées par la causalité. On peut ainsi s'affranchir par exemple des milliers d'opérations qui n'interfèrent pas avec les données qui nous intéressent. Enfin, elle s'avère également extrêmement précieuse pour simplifier la fusion des

données situées dans des branches différentes et mieux identifier les véritables conflits.

Un cas particulier d'opération d'évolution concerne l'évolution du schéma qui consiste à faire évoluer la structure des données sans perte d'information [Roddick93 - Publication « A taxonomy for schema versioning based on the relational and entity relationship models » Roddick, J.F., Craske, N.G. and Richards, T.J. 1993.]. De façon analogue aux données, le suivi de l'évolution de leur structure sera mieux assuré si le mécanisme de versionement de suivi des opérations et des traces causales s'applique également aux informations décrivant la structure. Des mesures particulières d'organisation des données et des metadonnées [Publication « Extracting delta for incremental data warehouse maintenance » Ram P et al. Data Engineering, 2000.] seront nécessaires.

Un des objectifs de la présente invention est de proposer un procédé faiblement intrusif et progressif d'organisation d'une base de donnée numérique sous une forme traçable. Nous visons l'assurance des niveaux successifs de traçabilité décrits ci-dessus, sans pour autant imposer la refonte des applications existantes.

En d'autres termes, l'objectif poursuivi par l'invention est de fournir aux applications informatiques et à leurs utilisateurs la capacité de suivre de façon précise les données tout au long de leur évolution, en traçant leurs histoires de façon complète, aussi bien sur le plan individuel (versions intermédiaires et liens de succession) que sur le plan collectif (évènements déclencheurs et liens d'interdépendance dynamiques issus des interactions entre les versions des données), en la

positionnant dans le cadre cohérent de son déroulement originel.

Il s'agit donc de fournir des liens de causalité à un niveau élémentaire où l'on puisse suivre aisément le flux causal des transformations et vérifier la validité de chaque opération intermédiaire sous la base des données d'entrée, du traitement appliqué et des données résultantes, de telle façon que la reconstitution de tout état du passé soit immédiate.

De plus, le procédé selon l'invention utilise un cadre architectural flexible, aussi peu contraignant et intrusif que possible afin de fournir une très large applicabilité au procédé proposé et une aussi large compatibilité que possible avec les procédés de stockage et manipulation des données courantes.

Afin d'assurer le suivi de l'évolution d'une base de données dite « principale », le procédé selon l'invention permet de faire en sorte qu'elle représente non pas seulement un mais tous les états cohérents successifs et/ou alternatifs nécessaires du monde réel représenté dans son évolution, tout en préservant les propriétés ACID.

Dans ce but, l'architecture mise en œuvre pour l'invention est illustrée figure 2 et est constituée comme suit :

un journal (J) organisé sous la forme d'une « base de données interne d'historisation » constitué d'une table ou d'un ensemble de tables dédiées au suivi de l'évolution et basées sur un mode de stockage universel à schéma stable (indépendant de la représentation logique des

données applicatives) et particulièrement adapté à la reconstitution à la volée des données.

un moniteur de transactions (M) et d'évènements capable de détecter toute demande d'évolution de valeurs et de structure transmise à la base de données qui rajoute au fur et à mesure dans le journal dédié des entrées caractérisant l'évolution élémentaire des données (identité, attribut, valeur, événement déclencheur et dépendances dynamiques)

un module de reconstitution (R) à la volée de l'état de la base de données selon un événement cible ; le système est muni dans ce but d'un curseur (C) dédié à la sélection de l'état recherché.

cas particulier : dans certains cas, il peut s'avérer utile de matérialiser la vue de la base dite « courante » ou « principale » sous la forme des tables de structure spécialisée, par exemple pour permettre des performances élevées et une compatibilité totale avec des applications existantes (notamment afin de permettre l'usage des procédures stockées et autres déclencheurs ou triggers qu'une application peut exiger pour fonctionner correctement).

Optionnellement, l'architecture comprend également :

- un système de suivi de la conformité (SC) des applications avec les états de la base et de son schéma
- des outils d'inoculation (I) automatique dans les applications d'instructions dédiées au suivi des dépendances dynamiques (capture des flux de données)

Le journal (J) d'événements (ou la base de données interne d'historisation) est constitué

principalement d'une table à structure indépendante de celle des données applicatives. Les colonnes sont :

- un identifiant unique de l'enregistrement de la table logique concerné par la ligne de journal, appartenant à la clé principale
- un identifiant de l'attribut dans le schéma, ou 0 pour l'enregistrement lui-même, appartenant à la clé principale
- un identifiant universel d'événement, incrémenté automatiquement, appartenant également à la clé principale du journal et correspondant à l'état de la base principale
- un champ *valeur* dédié au stockage des valeurs

Le rôle du moniteur (M) est de détecter et d'interpréter correctement chaque demande d'évolution en rajoutant l'information correspondante dans le journal d'évènements (J).

Exemples d'évolution de valeur

	ID	Attribut	UEID	Valeur	Commentaire
- insertion d'enregistrement	110	0	52	53	ID table « client »
- mise à jour d'un attribut	110	1	853	1001	No du client
- mise à jour d'un attribut	110	2	854	"aaa"	Nom du client
- suppression d'un enregistrement	110	0	981	0	Code suppression

Dans le langage d'échange avec une base de données SQL, les trois premières lignes du tableau peuvent être l'effet de la requête suivante :

insert into Client (no_client, nom_client) values (1001, "aaa")

Une telle requête est traitée comme suit :

- analyse syntaxique (parsing) de la requête
- récupération depuis le schéma des identifiants pour la table client (53) ainsi que pour les attributs « no_client » (1) et « nom_client » (2)
- insertion des trois lignes dans le journal

La dernière ligne peut être obtenue par l'instruction suivante :

delete from Client where No_client=1001

Une telle requête est traitée comme suit :

- analyse syntaxique (parsing) de la requête
- récupération depuis le schéma des identifiants pour la table client (53) ainsi que pour l'attribut « no_client » (1).
- récupération de l'identifiant de l'enregistrement du journal ayant la valeur 1001 pour l'attribut no 1
- insertion dans le journal de la dernière ligne (en utilisant le code 0 pour la valeur).

Exemples d'évolution de schéma

create table Client (no_client int primary key)

Création d'une nouvelle table

ID	Attribut	UEID	Valeur	Commentaire
53	0	252	8	ID table des tables
53	1	253	« Client »	Nom de la table
54	0	254	9	ID table des attributs
54	1	255	« no_client »	Nom de l'attribut
54	2	256	Int	Domaine
54	3	257	PK	Clé primaire
54	4	258	53	ID table

Ajout d'un attribut

alter table Client drop column no_client

Suppression d'un attribut

54	0	278	0	Code suppression

drop table Client

Suppression d'une table

53	0	293	0	Code suppression

Autres cas : déplacement d'attribut

54	3	308	22	Mise à jour ID table

L'exemple décrit ci-dessus concerne un cas complexe, sans équivalence en une seule opération SQL. Un

outil de gestion interactive peut en revanche permettre de tirer un réel bénéfice de cette caractéristique.

Comme on peut le remarquer, chaque événement qui tend à modifier la base de données logique finit par créer une ou plusieurs entrées sous la forme de nouvelles lignes (ou enregistrements) dans le journal. Ceci garantit que rien ne se perd et que toute suppression ou mise à jour logique ne se traduit pas en une suppression physique. Ainsi, les données du passé peuvent être récupérées. Un des avantages de cette organisation est la constitution concurrente de vues comme les livres de comptes qui bloquent généralement l'accès en mise à jour d'autres utilisateurs.

Remarquons également l'uniformité de la structure de stockage des informations : les données sont stockées en effet de façon identique, qu'il s'agisse de l'évolution des valeurs ou de celle des structures. Ceci veut dire que du point de vue logique, il est possible de reconstituer aussi bien les tables logiques que leurs structures, sur la base d'un même mécanisme. Par ailleurs, le fait d'inclure le journal dans la même base de données que la base principale permet de garantir sa cohérence relative de par le mécanisme transactionnel assuré par le SGBD.

Le module de reconstitution (R) a en charge justement la restitution en format logique des données en fonction d'un paramètre de type événement, à partir du journal d'événements (J).

Par exemple, considérons que l'application souhaite obtenir les données de la table Client telle

qu'elle était juste lors de l'événement 854. Cela implique au préalable la sélection de l'événement 854 par le curseur d'événements (C). Par la suite, la requête *"select * from Client"* est transmise au SGBD mais transformée par le module (R) en une requête plus complexe, obtenue de la façon suivante :

- reconstitution du schéma correspondant : la requête porte sur la table Client ; le système doit donc vérifier l'existence de la table Client au moment historique positionné par l'événement cible et récupérer les attributs de cette table logique ; (une optimisation est possible en gardant le schéma en cache)
- récupération des enregistrements dont le champ Attribut = 0 créés et non supprimés « avant » l'événement correspondant à l'état cible, (valeur = 0 pour le code de suppression) et attachés à cette table. Dans le cas des alternatives, « avant » ne concerne que les événements situés sur la même branche.
- récupération de tous les enregistrements dont le champ Attribut <> 0 attachés aux précédents et antérieurs à l'évènement cible.
- réorganisation du flux de données restituées et regroupement par enregistrement logique, c'est-à-dire dans notre cas, par client.

Dans un mode de réalisation de l'invention, il est possible de faire des requêtes de modification sur des états passés de la base de données principale de façon à créer une arborescence des versions de la base de données traitée.

En plus des valeurs et des événements, le journal peut accueillir les invocations d'opérations. Cela peut être réalisé par la représentation des opérations sous

la forme de tables logiques, où chaque opération correspond à un nom de table logique et chaque argument correspond à un attribut logique. En appliquant ce schéma de correspondance, l'application peut envoyer au journal (par exemple, par l'intermédiaire d'une API : « Application Programming Interface », interface de programmation d'applications) les informations nécessaires à la traçabilité des appels d'opération de façon analogue à la manipulation des données logiques (mais cette tâche peut être automatisée et confiée à un post-processeur, au compilateur, au processeur ou encore à la machine virtuelle).

add (2, 8)

Invocation de l'opération Add avec les arguments 2 et 8	ID	Attribut	UEID	Valeur	Commentaire
57 est l'identificateur de l'opération « add »	62	0	401	57	ID opération « Add »
	62	1	402	2	Premier argument
62 est l'identificateur de cette invocation de l'opération « add »	62	2	403	8	Second argument
	62	999	404	10	Valeur de retour

Les appels d'opération permettent de raccorder la sémantique des actions de l'application aux événements enregistrés dans le journal. Comme nous le verrons plus tard, cela facilitera le positionnement du curseur sur des repères significatifs du point de vue de l'utilisateur.

De surcroît, les points de validation des transactions peuvent être tracés sous la forme d'opérations. En effet, il est recommandé que le curseur se positionne exactement sur ces points et non pas entre deux opérations d'une même transaction. La cohérence du résultat en dépend. En revanche, des applications comme les outils d'aide à la conception peuvent très bien bénéficier des états intermédiaires, réputés incohérents, dans un but explicatif, et bénéficier ainsi de mécanismes de type « transactions longues ».

Précisons enfin que les opérations sont reliées par des références (non-représentées dans les tableaux) vers les opérations parentes de telle sorte que l'on puisse tracer également leur appartenance à l'exécution d'une opération de plus haut niveau. Ainsi, il sera possible de reconstituer l'appartenance des opérations, depuis le niveau élémentaire des évènements et jusqu'au niveau des transactions, en passant par autant de niveaux d'invocation que nécessaire pour les applications.

L'invention concerne également la matérialisation des liens de causalité.

Le flux des dépendances causales doit être constitué dynamiquement par les opérations de lecture et mise à jour en respectant les règles suivantes :

La manipulation des données doit systématiquement considérer aux côtés des données lues leurs références d'origine et les transporter tout au long

du flux de données et contrôle. L'application doit donc prendre en charge cet aspect, en ajoutant à chaque instruction de manipulation son équivalent de transport de références, par exemple par l'intermédiaire d'une API. L'automatisation de cette tâche peut être réalisée par un post-processeur et/ou par des extensions du processeur ou de la machine virtuelle.

Lors de l'insertion d'une donnée physique, les références du flux l'ayant alimentée doivent être stockées sous la forme d'une liste d'éléments de type ID-attribut-UEID, aux côtés de l'attribut *valeur*, et ceci pour chaque enregistrement physique du journal. Le tableau suivant en fait l'illustration. Une liste vide correspondrait à l'introduction d'une valeur de l'extérieur du système (par exemple, par la saisie effectuée par un utilisateur à travers une Interface-Homme-Machine).

ID	Attribut	UE ID	Valeur	Sources			Commentaire
110	2	543	"aaa"	...			
110	3	544	2	...			
...	...	...	...	...			
110	4	753	"aaa2"	ID	Attribut	UEID	La valeur de l'attribut 4 a été constituée à partir des attributs 2 et 3
				110	2	543	
				110	3	544	
...							

L'implémentation des sources dans le journal peut très bien être réalisée par l'intermédiaire d'un journal additionnel (ou sous-table), organisé de façon tabulaire, et ceci pour des raisons d'optimisation de performances, selon les techniques en vigueur dans la discipline des bases de données.

L'interprétation du flux se fait de manière simple : la valeur d'une donnée dépend des valeurs des données sources lues aux moments référencés par les événements UEID correspondants. On peut donc dire que les sources matérialisent les liens de causalité élémentaires.

L'invocation des opérations peut être tracée de la même manière. Voici à titre d'exemple, l'appel de l'opération Add (mentionnée précédemment) avec les arguments Client.Attr3 et la constante 7.

ID	Attribut	UEID	Valeur	Sources			Commentaire
62	0	401	57				ID opération « add »
62	1	402	2	ID	Attribut	UEID	Premier argument
				110	3	543	
62	2	403	7				Second argument
62	999	404	10				Valeur de retour

Le contrôle de validité des opérations peut être effectué par rapport aux données en vigueur. Par exemple, si la valeur de l'attribut Attr3 du Client 110 change après l'exécution de l'opération « add », les résultats envoyés par celle-ci ne peuvent plus être considérés comme conformes. On dit qu'il y a « remise en cause ». Dans le cas d'une évolution sans alternatives,

cela peut être vérifié grâce à une simple comparaison d'UEID entre les sources des arguments et les dernières valeurs des sources référencées.

Pour que cette information de traçabilité soit entièrement efficace pour l'utilisateur, il est utile de minimiser les constantes, c'est-à-dire les valeurs saisies « arbitrairement ». L'application doit ainsi privilégier des systèmes d'identification par liste de choix, par pointage, par glisser-déplacer, etc., ou par toute autre technique qui améliore à la fois l'ergonomie de l'application et qui permet implicitement l'assurance d'un suivi sans discontinuité du flux informationnel. En réalité, ces techniques d'identification sont largement répandues car elles assurent des avantages de référencement statique prévu dans les bases de données de façon courante.

Cette caractéristique du procédé permet de surcroît la mise en place d'un système d'optimisation automatique, qui - sur la base de la vérification systématique de la validité des sources - permet de renvoyer le résultat calculé précédemment, sans réexécuter effectivement l'opération. La mise en place d'une telle solution implique l'introduction des références vers les opérations appelantes (ce qui peut être fait à travers des arguments supplémentaires) et à condition que le temps de vérification soit inférieur à celui d'exécution (des statistiques de performances peuvent être maintenues à titre informatif et exploitées efficacement).

La notification automatique des « remises en cause » pourra être mise en place sur la base des informations de validité des versions des données par rapport aux flux. Ainsi, pour une opération, une classe

d'opération, une cible ou une source donnée, des alerteurs de cohérence de flux pourront notifier les applications par des messages synchrones ou asynchrones.

La ré-exécution consiste en une nouvelle invocation explicite d'une opération donnée sur le modèle d'une invocation précédente, mais sur la base de nouvelles valeurs. Dans tous les cas, elle donnera lieu à des nouvelles valeurs pour les données, les opérations et les sources tracées.

Le procédé selon l'invention est spécialement conçu pour gérer de façon opérationnelle l'historisation au fil de l'eau et la restauration à la volée. De plus, la gestion des volumes de stockage est facilitée et optimisée par un ensemble de facteurs :

- seules les valeurs attributs qui changent sont stockées (la redondance est ainsi minimisée)
- les volumes nécessaires de stockage supplémentaires augmentent de façon linéaire avec le nombre des attributs modifiés ou supprimés et ne dépendent pas des volumes de données insérés dans la base ; ce facteur autorise une utilisation très avantageuse pour un très large spectre d'applications.
- enfin, les purges très pertinentes peuvent être opérées selon les données marquées comme remises en cause par les liens de traçabilité de type source-destination, mais cette opération doit être pilotée par les applications en fonction de la sémantique des remises en cause.

Pour des raisons de simplification du discours, dans l'exemple précédent, nous avons fait l'hypothèse implicite d'une organisation séquentielle des événements et donc des états de la base principale (selon un ordre

total). Ainsi, pour vérifier la validité d'une source, nous avons évoqué comme solution la comparaison simple des identifiants universels d'événement (UEID).

En réalité, notre procédé autorise un vaste choix d'organisation des versions, comme par exemple :

- Arborescence : chaque événement a un événement parent. La valeur d'une donnée associée à un événement peut être obtenue par une remontée logique des parents jusqu'à la valeur la plus proche.
- Graphe orienté sans circuit : analogue à l'arborescence, cette organisation permet à une version d'avoir plusieurs parents différents. Les ambiguités de résolution peuvent être levées par des règles prédéfinies, basées sur des critères de priorité des branches ou sur tout autre caractéristique de la donnée (son type, etc.)

Les évolutions des branches différentes peuvent être fusionnées en faisant appel à la ré-exécution des opérations.

Les versions virtuelles sont des branches d'événements prédéfinies qui permettent la constitution de configurations parallèles pouvant bénéficier simultanément des événements appliqués à une ou plusieurs branches dites « de référence ». Autres caractéristiques :

- Les éventuels conflits sont évités par la séparation des événements par nature en branches de référence selon le modèle évoqué dans l'organisation de graphe orienté sans circuit.
- La matérialisation de ces configurations n'est pas réelle car les événements ne sont pas dupliqués physiquement (la propagation est logique).

L'architecture mise en œuvre pour la réalisation de l'invention peut aussi comporter les modules suivants

- un système de suivi de la conformité (SC) des applications avec les états de la base et de son schéma. Le principe est basé sur l'enregistrement d'un identifiant de version de l'application afin de déclarer un niveau de compatibilité avec le ou les états correspondants au schéma de la base principale
- des outils d'inoculation (I) automatique dans les applications d'instructions dédiées au suivi des dépendances dynamiques (capture des flux de données) : pré/post-processeur ou Machine virtuelle étendue
- des composants visuels spécialisés dans la navigation et l'exploration des états de la base (non-représentés).

L'invention peut être implémentée de plusieurs manières selon le contexte dans lequel elle est intégrée à une application.

La figure 3 présente une architecture qui autorise trois niveaux d'intégration de la traçabilité, de bas en haut :

Les applications existantes peuvent continuer à accéder à la base de données (dite « principale ») de la même manière. La base peut soit garder sa structure d'origine et rediriger les accès à un journal associé (dit base interne), soit évoluer vers une organisation physique de type journal et offrir des vues ou un driver ayant en charge la translation des requêtes et des résultats.

Les applications existantes pourront être très facilement munies d'un « curseur » à condition que l'accès aux données soit centralisé (ce qui est généralement le cas, par exemple à travers un driver unique). Dans ce cas, l'application pourra offrir des moyens d'accès automatiques

aux données de la base (implémentée désormais sous la forme d'un journal) et permettre aux utilisateurs d'actionner un curseur qui positionnera les lectures sur le repère événementiel désiré. Des légères adaptations pourront avoir lieu afin d'accorder la granularité des événements avec la sémantique de l'application.

Les nouvelles applications, entièrement construites sur la base des technologies d'inoculation de génération de traces bénéficieront implicitement du niveau le plus avancé de traçabilité offert par ce procédé comprenant le suivi exhaustif de l'évolution des données et de leur structure. Pour que le suivi de l'évolution des applications soit assuré au même niveau, il suffira de recourir à des techniques déclaratives de représentation des sources, de les confier au même journal et de les faire manipuler par un outil d'assemblage muni lui-même d'un module de traçabilité selon ce même procédé.

Cette architecture permet d'atteindre graduellement des niveaux de traçabilité des données persistantes de plus en plus élevées :

- initial : représentation et persistance (indispensable, préalable), assuré par le système initial de persistance
- journalisation des évènements (utile, reprise sur panne à court terme, mais pose un problème de reconstitution rapide des états passés)
- historisation et versionnement (utile, car les valeurs stockées sont multiples et peuvent comporter des variantes mais cette fonctionnalité génère des problèmes de reconstitution en mode compatible avec le mode initial)
- évolution structurelle : le suivi des évolutions des données et du schéma de la base de données principale, compatible avec le mode initial

- dépendance causale : la détection des flux de dépendance dynamique et des liens de causalité entre les données de la base de données d'historisation (journalisée)

L'utilisation de branches offre la possibilité de créer des alternatives d'évolution de la base de données. Dans le même temps, cela lève des nouveaux problèmes quant à la traçabilité. En effet, supposant qu'après la séparation des branches A et B, la donnée X soit modifiée dans la branche A à travers l'opération O. On peut alors souhaiter renvoyer sa nouvelle valeur dans la branche B, comme si elle avait eu cette valeur au moment de la séparation des branches. Cette opération, appelée rafraîchissement est très utile pour des nombreux cas où des données institutionnelles de référence sont reçues à des intervalles plus ou moins réguliers. Leur intégration peut poser alors des problèmes d'interférence avec les opérations effectuées entre temps. Par exemple, si aucune opération ayant eu comme source ou destination la donnée X dans la branche B n'a pas été effectuée entre temps, on peut sereinement considérer qu'il n'y a pas d'impact. En revanche, si c'est le cas, il faudra alors décider (explicitement ou implicitement) quelle est l'opération qui est prioritaire et refaire les autres. Ces conflits sont aisément détectables grâce aux liens de dépendance dynamique. La sémantique associée sera apportée quant à elle par celle des opérations ayant provoqué ces dépendances. La simple comparaison de l'identifiant universel des traces d'opérations permet d'évaluer l'antériorité et de la confirmer ou de l'infirmer. L'utilisateur (ou l'application, à travers un système de règles prédéfinies) peut ainsi trancher en connaissance de cause. Le cas de la fusion de branches est tout à fait analogue.

Précisons que cette technique est plus intéressante que le verrouillage anticipé d'une donnée, puisque, dans des nombreux cas, les opérations à venir ne sont pas prévisibles et leurs données cibles encore moins. La possibilité de créer des branches est d'ailleurs un moyen qui vise l'évitement au moins temporaire des conflits et qui permet de repousser à plus tard leur résolution.

Les branches virtuelles - qui sont par définition rafraîchies en permanence par leurs branches « parentes » - bénéficient automatiquement du rafraîchissement des données dans leurs branches parentes, y compris des opérations de débranchement (création de nouvelles branches) qui s'opèrent (virtuellement, bien entendu) en même temps sur les branches virtuelles. Par exemple, si la branche B est virtuelle, alors toute opération effectuée sur la branche A se répercute automatiquement sur la branche B. De plus, si l'on crée une nouvelle branche A2 à partir de A, celle-ci aura comme effet la création d'une sous-branche analogue B2 à partir de B. Il est important de souligner le caractère virtuel de ces rafraîchissements. Cela veut dire qu'en réalité aucun traitement n'est réellement effectué. Le seul effet réside dans le fait qu'une prochaine requête sur la branche B aura un résultat enrichi (qui tient compte des données rafraîchies). Précisons enfin qu'en cas de propagation automatique, il n'y a pas de résolution automatique de conflit, sauf si des règles ont été prédéfinies. Dans certains cas, on peut décider à l'avance que, par défaut, ce qui a été modifié explicitement dans la branche virtuelle reste prioritaire par rapport aux données provenues par rafraîchissement.

La fusion de données complexes est un cas à la fois plus sophistiqué et plus réaliste puisque le plus

souvent, le critère de décision majeur du choix de versions en vue de la résolution du conflit est le contexte. Considérons que la donnée X est une commande et que les données Y1 et Y2 sont deux de ses lignes de commande. Si un nouveau tarif pour l'article Z1 est proposé dans la branche « parente », puis propagé dans la branche en question, on doit alors décider si celui-ci remet en cause la valeur de la commande X, sachant que la ligne Y1 fait référence justement à l'article Z1. La réponse sera donnée par la règle de gestion en vigueur pour les commandes. Une telle règle pourrait s'exprimer par exemple sous la forme suivante : « si la commande se trouve dans l'état payé, la commande reste intacte, autrement, mes mises à jours tarifaires s'appliquent aussitôt ». Précisons que cette règle n'a pas à tenir compte des notions de version, branche ou encore de trace causale, ce qui souligne une nouvelle fois le très faible niveau d'intrusion de notre procédé.

En conclusion, la disponibilité des traces causales permet de configurer de façon plus fine les différentes possibilités de fusion, tout en respectant scrupuleusement les processus et tout en apportant la preuve irréfutable de ce respect.

Le spectre des applications de l'invention couvre la plupart des cas où il est utile de suivre l'évolution des données persistantes, des applications de gestion et jusqu'aux systèmes de gestion de fichiers, en passant par des outils de conception reposant sur des référentiels (ou repository), ou au-delà des besoins de persistance, pour peu que le suivi de l'évolution est utile.

L'invention est décrite dans ce qui précède à titre d'exemple. Il est entendu que l'homme du métier est à même de réaliser différentes variantes de l'invention sans pour autant sortir du cadre du brevet.

REVENDICATIONS

1. Procédé d'organisation d'une base de données numériques sous une forme traçable, comportant des étapes de modification d'une base de données numériques principale par ajout ou suppression ou modification d'un enregistrement de la base principale et des étapes de lecture de la base de données principale,

caractérisé en ce que

l'étape de modification de la base de données principale comprend une opération de création d'au moins un enregistrement numérique comportant au moins :

les identifiants numériques uniques des enregistrements et des attributs concernés de la base de données principale,

un identifiant numérique unique de l'état de la base de données principale correspondant à ladite modification de la base de données principale,

les valeurs élémentaires des attributs qui leur sont affectées à travers les opérations élémentaires, sans procéder au stockage des attributs ou des enregistrements non modifiés,

et d'ajout dudit enregistrement dans une base d'historisation interne composée d'au moins une table,

et en ce que l'étape de lecture portant sur tout état final ou antérieur de la base de données principale consiste à recevoir (ou intercepter) une requête originelle associée à l'identificateur unique de l'état visé, à procéder à une transformation de ladite requête originelle pour construire une requête modifiée d'adressage de la base d'historisation comprenant les critères de la requête originelle et l'identificateur de l'état visé, et de reconstruction du ou des enregistrements correspondant aux critères de la requête originelle et à l'état visé,

ladite étape de reconstitution consistant à retrouver les valeurs élémentaires, contenues dans les enregistrements de la base d'historisation, correspondant aux critères de la requête originelle [afin de réduire les besoins de capacité de stockage et les temps de traitement].

2. Procédé d'organisation d'une base de données numérique sous une forme traçable selon la revendication 1, caractérisé en ce que lesdits enregistrements de la base de données d'historisation contiennent également des références à d'autres enregistrements de la base de données interne, dans le but de préciser les liens de dépendance dynamique de type source-destination constituant le flux causal des interférences entre les versions des données

3. Procédé d'organisation d'une base de données numérique sous une forme traçable selon l'une quelconque des revendications précédentes, caractérisé en ce que

ladite opération de modification de la base principale est une opération logique et que

ladite opération d'ajout dans la base de données d'historisation consiste à ajouter :

un enregistrement identifiant l'état de la base correspondant à l'opération logique,

autant d'enregistrements que de paramètres de l'opération logique,

un enregistrement pour le résultat éventuel de l'opération logique

et à préciser par un lien de parenté les regroupements d'opérations depuis le niveau élémentaire de modification jusqu'au niveau de la transaction, en passant le nombre de niveaux sémantiques nécessaires pour les applications.

4. Procédé d'organisation d'une base de données numérique sous une forme traçable selon l'une quelconque des revendications précédentes, caractérisé en ce que la base de données principale contient une ou plusieurs table(s), organisant les liens d'évolution entre les identifiants des états successifs et alternatifs de la base principale, destinée(s) à organiser les enregistrements de la base de données interne.

5. Procédé d'organisation d'une base de données numérique sous une forme traçable selon la revendication 4, caractérisé en ce que ladite ou lesdites tables des liens d'évolution entre les états de la base principale contiennent des enregistrements spécifiant les règles de correspondance entre les enregistrements de la base de données interne d'historisation et les états de la base de données principale.

6. Procédé d'organisation d'une base de données numérique sous une forme traçable selon l'une des revendications 4 à 5, caractérisé en ce que ladite opération de lecture consiste à déterminer ledit état de la base de données principale en se référant aux dits identifiants et aux tables des liens d'évolution entre les états de la base principale.

7. Architecture de gestion de base de données utilisant le procédé d'interrogation de l'une quelconque des revendications précédentes, caractérisée en ce qu'une application interrogeant la base de données principale peut spécifier l'état de la base de données principale désiré.

8. Architecture de gestion de base de données selon la revendication 7, caractérisée en ce que ladite

application peut opérer des modifications sur tout état de la base principale et donnant lieu, dans le cas de la tentative de modification d'un état antérieur, à la création de nouvelles alternatives d'évolution de la base de données principale dont les données seront générées par la même base d'historisation interne.

9. Procédé d'organisation d'une base de données numérique sous une forme traçable selon l'une quelconque des revendications 3 à 6, caractérisé en ce que les liens de dépendances servent de critères de remise en cause desdites opérations déjà effectuées.

10. Procédé d'organisation d'une base de données numérique sous une forme traçable selon l'une quelconque des revendications 3 à 6, caractérisé en ce que les mises à jour effectuées sur des branches différentes pourront être intégrées ou fusionnées dans le cadre d'un nouvel état « héritant » desdites branches.

11. Procédé d'organisation d'une base de données numérique sous une forme traçable selon l'une quelconque des revendications 3 à 6, caractérisé en ce que les cas d'évolution de structure des données de la base de données principale sont traités comme des cas particuliers d'évolution des données de ladite base, pour peu que la structure/schéma de ladite base principale soit décrite de la façon mentionnée pour les données, en tant que dictionnaire.

12. Procédé d'organisation d'une base de données numérique sous une forme traçable selon l'une quelconque des revendications 3 à 6, caractérisé en ce que la base de données d'historisation est explorée et

interrogée par des applications à travers le mode natif du SGBD afin d'obtenir des informations comme par exemple toutes les valeurs historiques d'un attribut et toutes les incidences (dynamiques) de toute mise à jour et de naviguer au long des versions et des flux de dépendance dynamique et ceci de façon classique, selon le langage d'interrogation en vigueur, exigé par le SGBD.

Figure 1

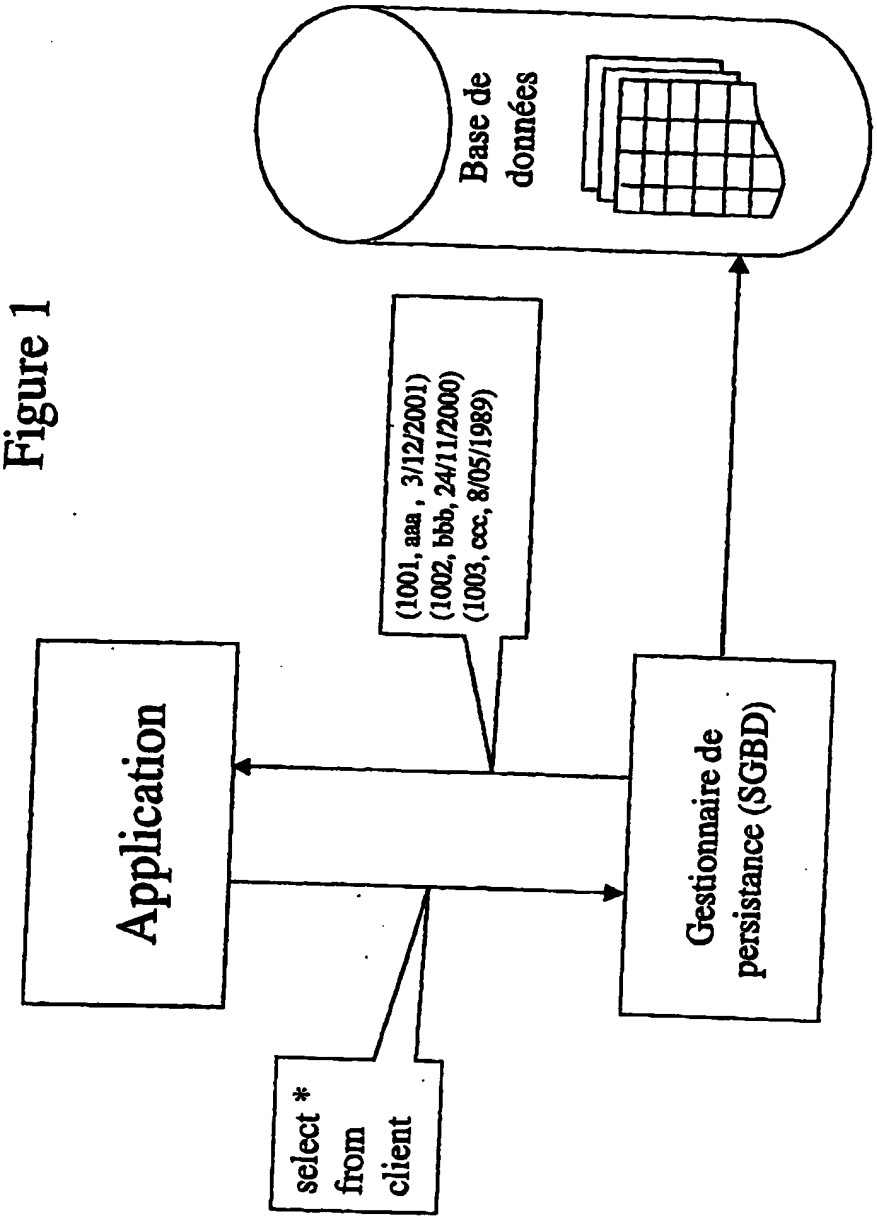
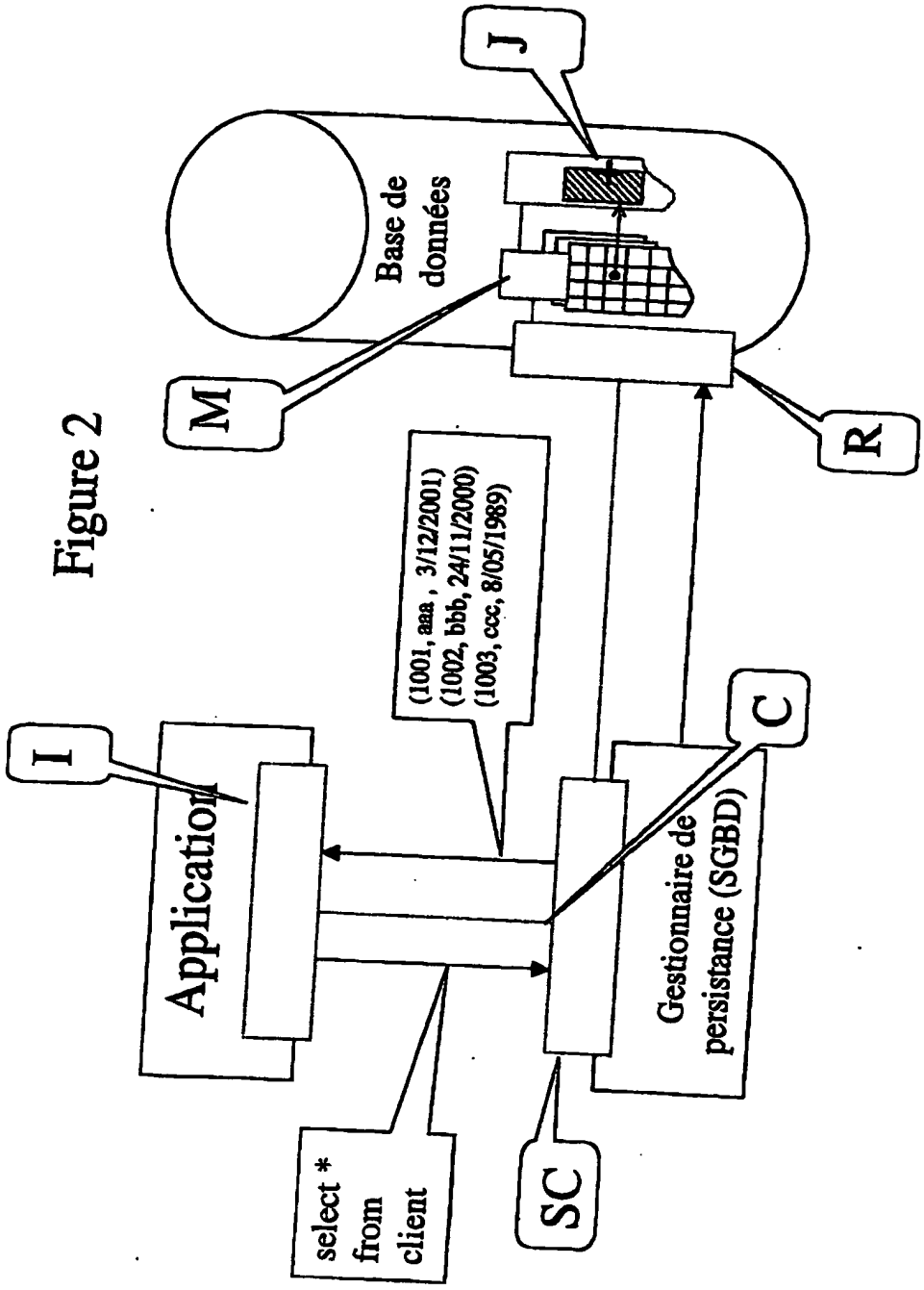

Base de données
(1001, aaa , 3/12/2001)
(1002, bbb, 24/11/2000)
(1003, ccc, 8/05/1989)
Application
Gestionnaire de persistance (SGBD)
select * from client

Figure 2


J
Base de données
M
R
(1001, aaa , 3/12/2001)
(1002, bbb, 24/11/2000)
(1003, ccc, 8/05/1989)
I
Application
C
Gestionnaire de persistance (SGBD)
select * from client
SC

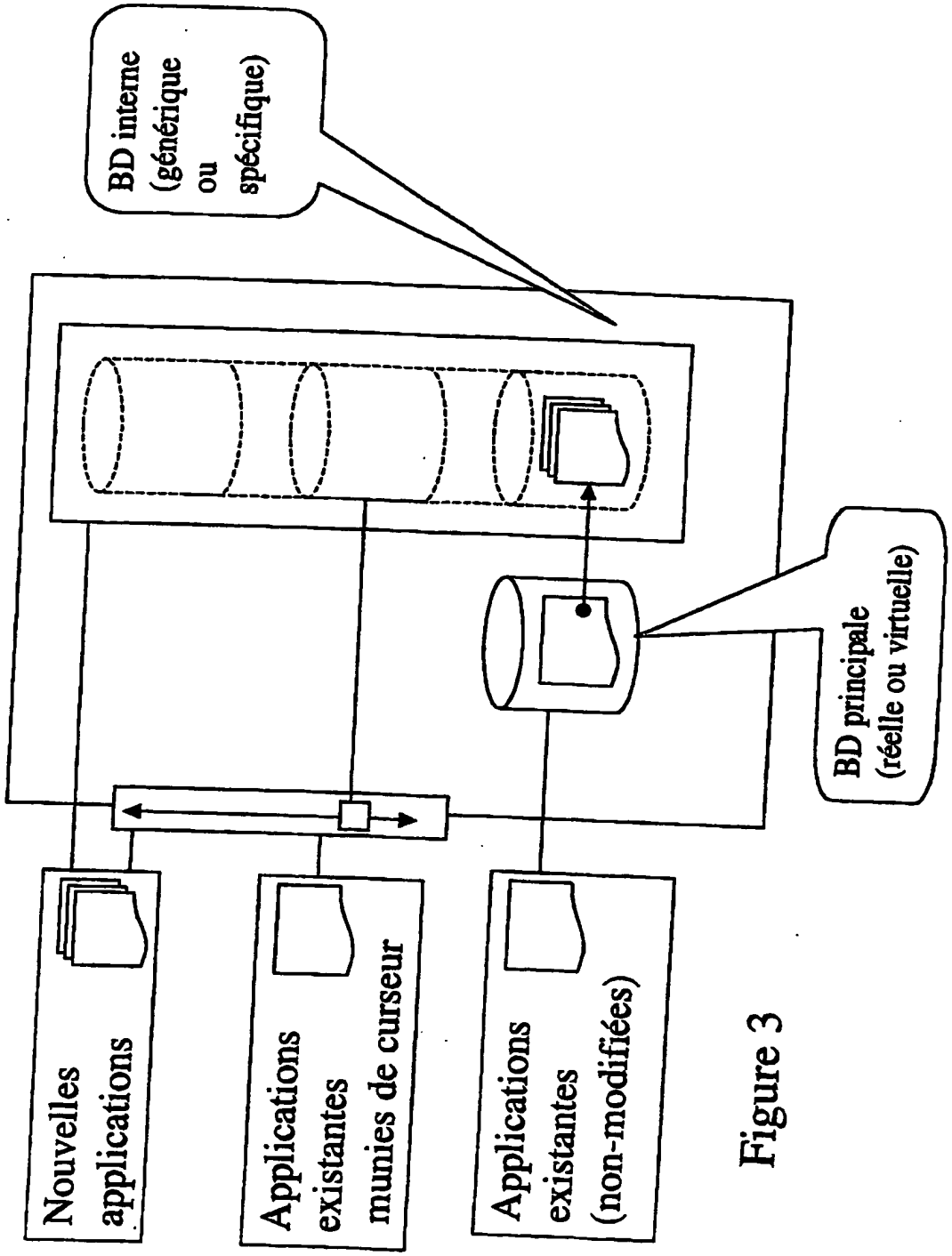

BD interne (générique ou spécifique)
BD principale (réelle ou virtuelle)
Nouvelles applications
Applications existantes munies de curseur
Applications existantes (non-modifiées)

Figure 3